

INSTITUTO TECNOLÓGICO DE AERONÁUTICA



Geune Vieira Quintino

**ACOPLAMENTO DA ANÁLISE DE ÁRVORE DE
FALHAS EM UMA ARQUITETURA SISTÊMICA**

Trabalho de Graduação
2021

Curso de Engenharia Aeroespacial

Geune Vieira Quintino

**ACOPLAMENTO DA ANÁLISE DE ÁRVORE DE
FALHAS EM UMA ARQUITETURA SISTÊMICA**

Orientador

Prof. Dr. Christopher Shneider Cerqueira (ITA)

ENGENHARIA AEROESPACIAL

SÃO JOSÉ DOS CAMPOS
INSTITUTO TECNOLÓGICO DE AERONÁUTICA

Dados Internacionais de Catalogação-na-Publicação (CIP)
Divisão de Informação e Documentação

Quintino, Geune Vieira
Acoplamento da Análise de Árvore de Falhas em uma arquitetura sistêmica / Geune Vieira
Quintino.
São José dos Campos, 2021.
46f.

Trabalho de Graduação – Curso de Engenharia Aeroespacial– Instituto Tecnológico de
Aeronáutica, 2021. Orientador: Prof. Dr. Christopher Shneider Cerqueira.

1. MBSE. 2. Capella. 3. FTA. I. Instituto Tecnológico de Aeronáutica. II. Título.

REFERÊNCIA BIBLIOGRÁFICA

QUINTINO, Geune Vieira. **Acoplamento da Análise de Árvore de Falhas em uma arquitetura sistêmica**. 2021. 46f. Trabalho de Conclusão de Curso (Graduação) – Instituto Tecnológico de Aeronáutica, São José dos Campos.

CESSÃO DE DIREITOS

NOME DO AUTOR: Geune Vieira Quintino

TÍTULO DO TRABALHO: Acoplamento da Análise de Árvore de Falhas em uma arquitetura sistêmica.

TIPO DO TRABALHO/ANO: Trabalho de Conclusão de Curso (Graduação) / 2021

É concedida ao Instituto Tecnológico de Aeronáutica permissão para reproduzir cópias deste trabalho de graduação e para emprestar ou vender cópias somente para propósitos acadêmicos e científicos. O autor reserva outros direitos de publicação e nenhuma parte deste trabalho de graduação pode ser reproduzida sem a autorização do autor.



Geune Vieira Quintino

Rua H8A, 121

12.228-460 – São José dos Campos–SP

ACOPLAMENTO DA ANÁLISE DE ÁRVORE DE FALHAS EM UMA ARQUITETURA SISTÊMICA

Essa publicação foi aceita como Relatório Final de Trabalho de Graduação



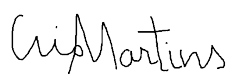
Geune Vieira Quintino

Autor



Christopher Shneider Cerqueira (ITA)

Orientador



Prof^a. Dr^a. Cristiane Aparecida Martins
Coordenadora do Curso de Engenharia Aeroespacial

São José dos Campos, 22 de dezembro de 2021.

A todos aqueles que acreditaram em mim.

Agradecimentos

Primeiramente, gostaria de agradecer a Deus, por estar comigo e por ter me guiado e acompanhado ao longo de toda a jornada até aqui.

Aos meus pais, Esmeralda Vieira Santana e João Quintino Lucas, pelo dom da vida, pelo amor, pelos ensinamentos e cobranças, e por tudo que sou e serei na vida.

À minha tia Jacinta Vieira Neta, pelo empenho, pela confiança, pelo afeto e pelos ensinamentos, os quais foram imprescindíveis para a concretização deste objetivo.

Aos meus irmãos, João Emanuel Vieira Quintino, João Quintino Lucas Júnior e Daniel Vieira Quintino, pela confiança depositada e pela dedicação e suporte que me deram para tornar possível este sonho.

Aos amigos Fidel Esteves do Nascimento, Daniel Chin e Caio Felipe Siqueira Gomes, pelos momentos compartilhados, pelos ensinamentos, pelo companheirismo e por tornar toda a jornada mais leve.

Ao Prof. Dr. Christopher Shneider Cerqueira, pela orientação, pela confiança e pela colaboração empreendida na realização deste trabalho.

À AESP 20, pela união de turma, e, em especial, aos colegas e amigos do Viradão Team.

Ao Colégio 7 de Setembro, pelos ensinamentos, pela orientação, pela confiança e suporte depositados em mim.

“A imaginação é mais importante que o conhecimento.”

— ALBERT EINSTEIN

Resumo

Este trabalho irá explorar a utilização da Análise de Árvore de Falhas para avaliação de segurança e confiabilidade de arquiteturas construídas em softwares de Engenharia de Sistemas, de forma a explicitar a robustez de uma arquitetura de um sistema complexo. Os sistemas complexos requerem ferramentas de exploração e gerenciamento de sua complexidade para possibilitar aos engenheiros: a visão do domínio do problema identificando as necessidades dos *stakeholders* e seus requisitos; a visão do domínio da solução objetivando identificar alternativas de arquiteturas que possam vir a se tornar soluções candidatas à concretização do projeto. Com o auxílio dessa análise integrada, será possível construir a arquitetura em uma ferramenta e explorar a análise de risco no modelo e suas implicações.

Abstract

This work will explore the use of Fault Tree Analysis to assess the security and reliability of architectures built on Systems Engineering software in order to explain the robustness of a complex system architecture. Complex systems require tools to explore and manage their complexity to enable engineers: a view of the problem domain identifying the needs of *stakeholders* and their requirements; a view of the solution domain aiming to identify alternative architectures that may become candidate solutions for the project's implementation. With the help of this integrated analysis, it will be possible to build the architecture in a tool and explore the risk analysis in the model and its implications.

Sumário

1	INTRODUÇÃO	11
1.1	Objetivo	11
1.2	Motivação	11
2	REVISÃO BIBLIOGRÁFICA	13
2.1	Engenharia de Sistemas	13
2.2	Engenharia de Sistemas Baseada em Modelos (MBSE)	14
2.3	Metodologia ARCADIA	14
2.4	Kitalpha	16
2.5	Capella	17
2.6	Análise da Árvore de Falhas	17
3	MATERIAIS E MÉTODOS	20
3.1	Plano de Atividades	20
3.2	Desenvolvimento do <i>viewpoint</i>	20
4	RESULTADOS E DISCUSSÃO	23
4.1	Desenvolvimento do Viewpoint e Integração MBSE	24
4.2	Interface Gráfica	25
4.3	Aplicações	26
4.3.1	Explosão da Câmara de Combustão	26
4.3.2	Perda de Navegação	27
4.3.3	Falha da Missão	27
5	CONCLUSÃO	29

REFERÊNCIAS	30
APÊNDICE A – CÓDIGOS DO VIEWPOINT	32
A.1 Aspects	32
A.2 Data	33
A.3 Diagram	35
A.4 UI	41
A.5 Services	44
A.6 Configuration	45
A.7 Build	46

1 Introdução

1.1 Objetivo

O objetivo principal deste trabalho é desenvolver a Análise de Árvore de Falhas (do inglês *Fault Tree Analysis*, FTA) através de diagramas construídos nos moldes da norma internacional IEC 61025 (COMMITTEE *et al.*, 2006) para aplicação em projetos aeroespaciais. O objetivo secundário é implementar esses diagramas em *viewpoints* no ambiente de desenvolvimento Kitalpha e integrá-los à ferramenta Capella.

1.2 Motivação

Com o rápido avanço da Engenharia e da Tecnologia nas últimas décadas, sistemas complexos têm surgido em praticamente todas as áreas de atuação humana. Com isso, são cada vez mais necessárias técnicas de alto desempenho para o desenho, arquitetura, desenvolvimento e gestão desses sistemas. O uso de Engenharia de Sistemas Baseada em Modelos (do inglês *Model-Based Systems Engineering*, MBSE) tem se mostrado como técnica promissora para esse propósito, principalmente por sua simplicidade, consistência e padronização de uso.

A análise de confiabilidade e segurança desses sistemas é fundamental para garantir que o sistema atenda aos requisitos funcionais e operacionais dos *stakeholders*. A detecção antecipada de eventos indesejados e suas causas possibilitam o controle desses eventos e evitam o trabalho custoso e demorado caso o sistema viesse a falhar por conta deles. Dessa forma, o presente trabalho busca desenvolver e analisar a Árvore de Falhas dos componentes do sistema e determinar as causas de eventos indesejados (falhas) e estimar a probabilidade de ocorrência desses eventos.

Para isso, o trabalho buscará, seguindo a norma internacional IEC 61025 (COMMITTEE *et al.*, 2006), desenvolver um *framework* para o diagrama dessas árvores de falhas utilizando a ferramenta Eclipse Kitalpha e por fim integrá-lo na ferramenta Capella. Dessa forma, a principal contribuição deste trabalho é a extensão da Análise de Árvore de Falhas para

o domínio MBSE e a posterior integração do uso na ferramenta Capella.

2 Revisão Bibliográfica

2.1 Engenharia de Sistemas

O conceito de “sistema” remete a um conjunto de elementos que interagem conjuntamente para a realização de um objetivo pré-estabelecido. Em termos gerais, “Engenharia” pode ser entendida como um conjunto de atividades que se utiliza de conceitos, metodologias e ferramentas científicas para conceituar, desenhar, elaborar e manter: sistemas, processos e produtos. Nesse sentido, pode-se conceituar Engenharia de Sistemas como um ramo multidisciplinar da engenharia focado no desenho, na arquitetura, no desenvolvimento e na gestão de sistemas.

(INCOSE..., 2015) define Engenharia de Sistemas como uma metodologia que ajuda na solução de problemas, seja ele qual for. Problema é definido como a diferença entre o estado real e o estado desejado. Através da investigação e pesquisa para o conhecimento e elaboração dos dois estados, Engenharia de Sistemas se apresenta como uma metodologia dinâmica para a redução contínua do delta entre eles. Os autores ponderam que apesar de sua importância, a metodologia não é o único fator e geralmente não é o mais importante para a solução de problemas. Outros componentes fundamentais incluem: conhecimento do especialista, conhecimento da situação, experiência, comportamento, psicologia e ética profissional.

Segundo (LIBRARY; ADMINISTRATION, 2017), Engenharia de Sistemas pode ser definida como uma maneira de se atingir os requisitos funcionais, físicos e operacionais dos *stakeholders* dentro do custo, cronograma e outras restrições (*constraints*), as vezes conflitantes. De outro modo, é uma maneira lógica de pensar. Além disso, os autores ponderam que é essencial ao Engenheiro de Sistemas a habilidade de identificar e focar esforços em avaliações para otimizar o projeto geral e não favorecer um sistema/subsistema às custas de outro, o que deve ser feito em paralelo com a constante validação dos objetivos operacionais. Essas habilidades são as que constituem o pensamento sistêmico.

2.2 Engenharia de Sistemas Baseada em Modelos (MBSE)

Segundo (HART, 2015), um modelo pode ser entendido como um versão simplificada de um conceito, fenômeno, relação ou estrutura de um sistema. Pode ser gráfico, matemático ou uma representação física. Ao eliminar componentes desnecessários de sistemas complexos, os modelos: possibilitam um maior entendimento sobre os componentes chaves do sistema; ajudam no processo de decisão sobre diferentes cenários; ajudam a explicar, controlar e prever eventos.

MBSE pode ser entendida como uma técnica que se baseia na aplicação de modelos para a Engenharia de Sistemas. Por ser uma disciplina multidisciplinar, assim como muitos outros ramos da ciência e da engenharia, Engenharia de Sistemas pode se beneficiar imensamente de modelos voltados a darem suporte em seus diversos campos de atuação, dentro dos quais destacam-se: definição dos requisitos do sistema, desenho, análises, verificação, validação, integração e testes.

(MCDERMOTT *et al.*, 2020) conduziu uma pesquisa com 240 participantes de várias organizações governamentais e indústrias para entender os benefícios e a efetividade na implementação da Engenharia Digital (*Digital Engineering*, DE) e da MBSE nas correspondentes organizações. Em paralelo, os autores realizaram separadamente uma revisão da literatura vigente para identificar os benefícios da adoção de DE/MBSE. A Tabela 2.1 apresenta os 48 benefícios identificados categorizados nos 4 grandes grupos: Qualidade, Velocidade/Agilidade, Experiência do Usuário e Transferência do Conhecimento. Os participantes da pesquisa citaram 45 dos 48 benefícios identificados na literatura. Tais resultados testificam que de fato modelos podem ser grandes aliados da Engenharia de Sistemas.

2.3 Metodologia ARCADIA

ARCADIA (*ARChitecture Analysis and Design Integrated Approach*) é um método utilizado para definir e validar a arquitetura de sistemas complexos. Criado em 2007 pela empresa francesa Thales, o método promove o trabalho colaborativo entre todos os *stakeholders* envolvidos no processo da engenharia de sistemas. Através do alinhamento e interação dos processos, esse método ajuda para que o *design* da arquitetura de sistemas atenda aos requisitos de projeto.

Segundo (ROQUES, 2018), os princípios gerais da metodologia ARCADIA são os seguintes:

- Todos os envolvidos no processo de engenharia de sistemas compartilham das mes-

TABELA 2.1 – Lista de benefícios da DE/MBSE (MCDERMOTT *et al.*, 2020).

Categoria	Lista de Benefícios		
Qualidade	Redução de erros/defeitos	Aprimoramento da análise de riscos	Capacidade aprimorada
	Rastreabilidade aprimorada	Projeto de sistema aprimorado	Maior envolvimento dos <i>stakeholders</i>
	Aprimoramento da qualidade do sistema	Melhor geração de requisitos	Testes reforçados
	Redução o risco	Maior precisão das estimativas	Redução do custo
	Maior rigor	Melhor capacidade preditiva	Melhor capacidade de análise
	Maior eficácia	Maior qualidade de entrega	
Velocidade/Agilidade	Maior consistência	Maior produtividade	Suporte de nível superior para integração
	Maior capacidade de reutilização	Maior transparência	Maior uniformidade
	Maior eficiência	Maior confiança	Maior precisão
	Redução do retrabalho	Maior flexibilidade	V&V antecipado
	Redução do tempo	Melhor gerenciamento de requisitos	Reduz a ambiguidade
Redução do desperdício	Facilidade de personalização de design	Fácil de fazer alterações	
Experiência do Usuário	Suporte de nível superior para automação	Melhor compreensão do sistema	Reduz esforços
	Redução da carga das tarefas da ES	Melhor gerenciamento/captação de dados	
	Gerenciar melhor a complexidade	Melhor tomada de decisões	
Transferência do Conhecimento	Melhor acessibilidade de informações	Arquitetura aprimorada	Colaboração aprimorada
	Melhor gestão/captura do conhecimento	Melhor compartilhamento de comunicação/informação	Vários pontos de vista do modelo

mas metodologias, informações, necessidades e descrições dos modelos de projeto.

- Cada processo da engenharia é formalizado através de um *viewpoint* em relação aos requisitos correspondentes a partir dos quais a arquitetura é verificada.
- As regras para a verificação da arquitetura são previamente estabelecidas para possibilitar a verificação o mais rápido possível.
- A engenharia conjunta dos diferentes níveis é apoiada pela comum elaboração de modelos, de modo que os modelos são deduzidos, validados e vinculados um com o outro.

Ainda segundo os autores, os níveis gerais de atuação da metodologia são:

- Análise Operacional: “O que os usuários do sistema precisam realizar?”
 - Análise dos problemas de problemas dos usuários operacionais e identificação das atividades e interações dos fatores do sistema.
 - Definição das necessidades operacionais dos *stakeholders*.
- Análise das Necessidades do Sistema: “O que o sistema tem que realizar para os usuários”

- Análise externa funcional como uma forma de identificar as necessidades do sistema para atender aos usuários.
- Formaliza os requisitos do sistema.
- Arquitetura Lógica: “Como o sistema funcionará para atender às expectativas”
 - Análise interna das subfunções do sistema a serem montadas e integradas para atender as necessidades identificadas pelo nível anterior.
 - Identificação dos componentes lógicos que realizam essas subfunções, integrando as restrições não-funcionais desse nível.
- Arquitetura Física: “Como ficará o sistema desenvolvido e construído”
 - Define a arquitetura final do sistema como deve ser criado;
 - Destaca as funções requeridas pela implementação e análise técnica e identifica os componentes e materiais necessários.
- Estrutura de decomposição do produto final (EPBS): “O que se espera de cada provedor do sistema?”
 - Seu utiliza das análises anterior da arquitetura do sistema para identificar as condições que cada componente deve satisfazer para atender aos requisitos e limitações das fases anteriores.

Os benefícios da implementação efetiva da metodologia têm sido observados pela Thales em seus muitos domínios de atuação (transporte, aviônica, espaço, radar, etc.). Tais benefícios incluem: consistência geral do projeto, acessibilidade da equipe a diferentes *viewpoints*, identificação precoce de possíveis problemas de design e análise de impactos em caso de mudanças de requisitos (CALIÒ *et al.*, 2016).

2.4 Kitalpha

Kitalpha é uma ferramenta para a implementação e modelagem de *viewpoints* e *frameworks* baseados na metodologia MBE (Engenharia Baseada em Modelos). Cada *viewpoint* é responsável por descrever uma parte da arquitetura do sistema, como performance, segurança e custo. O ambiente de desenvolvimento possui metamodelos e representações (tabelas e diagramas) que atendem e formalizam as diferentes especialidades da engenharia (KITALPHA, 2021).

2.5 Capella

Capella é uma solução de código aberto para a Engenharia de Sistemas Baseada em Modelos. Desenvolvida pela empresa francesa Thales em 2007, a ferramenta foi lançada em 2015 como um projeto de código aberto Eclipse pelo grupo francês PolarSys, um grupo de trabalho da Fundação Eclipse (Wikipedia contributors, 2021). Baseada na metodologia ARCADIA, Capella fornece o suporte metodológico e orientação para engenheiros de sistemas, softwares e hardwares. O projeto visa complementar a cultura de código aberto com uma comunidade que reúne os principais atores da toda a cadeia de valor da engenharia (industriais, integradores, fornecedores de tecnologia e consultores, academia) para inovação aberta dentro do Capella (BOUDJENNAH *et al.*, 2015).

Capella fornece uma série de diagramas para modelagem gráfica, dentre os quais destacam-se:

- Análise Operacional: desenvolve a análise operacional.
- Análise de Sistema: desenvolve a análise das necessidades do sistema.
- Arquitetura Lógica: desenvolve a arquitetura lógica do sistema.
- Arquitetura Física: desenvolve a arquitetura física do sistema.
- EDPF: desenvolve a estrutura de decomposição do produto final.

2.6 Análise da Árvore de Falhas

A Análise da Árvore de Falhas (FTA) é uma abordagem sistemática *top-down* para a análise da confiabilidade e segurança de sistemas. Através da análise da propagação de falhas pelo sistema, esse método possibilita determinar as causas de eventos indesejados (análise qualitativa) bem como calcular a probabilidade de ocorrência desses eventos (análise quantitativa), o que permite avaliar se o sistema está ou não em conformidade com os requisitos de segurança e confiabilidade de projeto. Cada Árvore de Falha é responsável por analisar um único evento indesejado, denominado evento principal (*top event*).











Os principais elementos no modelo de árvore de falha são Eventos, Ramos e Portas Lógicas. Cada ramo está associado a um evento e a uma porta lógica. A multiplicidade reflete situações onde um evento (ou porta lógica) pode ser associado a um ou vários ramos. A Tabela 2.2 mostra os principais símbolos utilizados na árvore de falha de acordo com a norma internacional IEC 61025 (COMMITTEE *et al.*, 2006).

Os Eventos consistem do Evento Intermediário ou de Saída, Evento Primário, Evento de Transferência e Evento Condicional, cada um representando o tipo de evento correspondente visto nas Árvores de Falhas. O Evento Primário generaliza o Evento Básico, Evento não Desenvolvido e Evento Externo. O Evento de Transferência representa a existência de ramificações externas à árvore.

Portas Lógicas consistem da Porta AND, Porta AND Prioritária, Porta OR, e Porta de Inibição a qual é usada para avaliar os ramos de entrada e amarrar os ramos juntos.

Evento Condicional está associado com a Porta AND Prioritária ou com a Porta de Inibição. Ele estabelece uma condição necessária para a Porta Lógica ocorrer. Essas portas são sempre acompanhadas do Evento Condicional especificado em um forma oval.

TABELA 2.2 – Símbolos da Árvore de Falhas.

Símbolo	Descrição
	Evento Intermediário ou de Saída - evento que é uma saída de um símbolo lógico (referido como evento principal ou evento intermediário)
	Evento Básico - evento de nível mais baixo para o qual a probabilidade de ocorrência ou informações de confiabilidade estão disponíveis
	Evento Externo - evento que é externo a um sistema
	Evento não Desenvolvido - evento primário que representa uma parte do sistema que ainda não está desenvolvida
	Evento Condicional - evento que é uma condição de ocorrência de outro evento quando ambos têm que ocorrer para a saída ocorrer
	Evento de Transferência - evento que indica que determinada parte do sistema é desenvolvida em outra parte ou página do diagrama
	Porta AND - Porta lógica booleana - o evento de saída ocorre apenas se todos os eventos de entrada ocorrerem
	Porta OR - Porta lógica booleana - o evento de saída ocorre se pelo menos um evento de entrada ocorrer
	Porta AND Prioritária - Porta lógica booleana - o evento de saída (falha) ocorre apenas se todas as entradas eventos ocorrerem em sequência da esquerda para a direita
	Porta de Inibição - Porta - Porta lógica booleana - o evento de saída ocorre apenas se ambos os eventos de entrada ocorrerem e um deles for condicional

De acordo com (Wikipedia contributors, 2020), a Análise da Árvore de Falhas abrange as 5 etapas:

- Definição do evento indesejado
 - O Evento indesejado pode ser muito difícil de definir. Cabe ao engenheiro com a melhor experiência no sistema em estudo definir e numerar os eventos indesejados.
- Compreensão do sistema.
 - Após a definição do evento indesejado, todas as causas que podem afetar esse evento precisam ser estudadas e analisadas.
- Construção da árvore de falhas.
 - Após definido o evento indesejado e os efeitos causadores, pode-se começar a construir a árvore de falhas, a qual se baseia na integração desses componentes com os ramos e portas lógicas.
- Avaliação da árvore de falhas.
 - Nesta etapa é feita uma avaliação da árvore obtida. Análises qualitativas e quantitativas são feitas para identificar todas as possíveis causas que levam o sistema à falha bem como a probabilidade estimada de ocorrência do evento indesejado.
- Controle os perigos identificados.
 - Após a identificação de possíveis combinações de eventos que levam o sistema a falha, são tomadas medidas para diminuir a probabilidade estimada.

3 Materiais e Métodos

3.1 Plano de Atividades

Inicialmente, os componentes da árvore de falhas (eventos, ramos e portas lógicas) são definidos a partir da norma internacional IEC 61025. Esses componentes são então usados na implementação da arquitetura da árvore em *viewpoints* no ambiente de desenvolvimento kitalpha, onde são desenvolvidos os diagramas da árvore e os serviços a serem disponibilizados aos usuários. Os *viewpoints* contruídos são então integrados à ferramenta Eclipse Capella 5.0.0, os quais passarão a estar disponíveis para a modelagem e análise gráfica das arquiteturas de sistemas formalizados dentro da ferramenta. Por fim, a integração proposta será utilizada para a construção de FTA de projetos no contexto do Centro Espacial ITA (CEI). A Figura 3.1 resume o plano de atividades proposto.

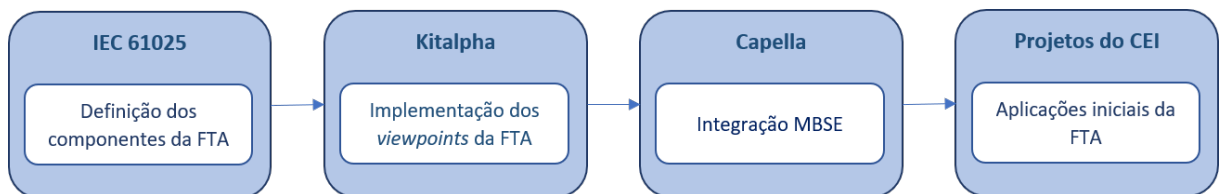


FIGURA 3.1 – Plano de atividades.

3.2 Desenvolvimento do *viewpoint*

Para a criação do *viewpoint* da FTA, utilizou-se a ferramenta Eclipse Capella Studio 5.0.0, que baseado na tecnologia do *Kitalpha* e do Capella, fornece as ferramentas necessárias para o desenvolvimento do *viewpoint* e a capacidade de estendê-lo ao Capella. Para isso, com o Capella Studio aberto, seleciona-se a opção de projeto *Viewpoint DSL Project* através do caminho *File* → *New* → *Project* → *Kitalpha* → *Architecture description* → *Viewpoint DSL project* e, em seguida, entra-se com o nome do *viewpoint*, FTA neste caso, e seleciona-se o Capella como a aplicação de destino (*target application*). O diagrama será estendido para o nível Arquitetura Física (*PhysicalArchitecture*) do Capella, o qual poderá ser integrado e utilizado nesse contexto. A estrutura DSL (Linguagem de Domínio Espe-

cífico, do inglês *Domain-specific language*) garante que cada *viewpoint* tenha seu próprio domínio de linguagem, bem como comandos específicos. Com isso, o desenvolvimento de um projeto *Viewpoint DSL* se dá por uma linguagem e comandos próprios, embora possa-se identificar uma semelhança com outras linguagens em determinados elementos lógicos e com a sintaxe, como o Java.

Após esse processo inicial, a estrutura do projeto é criada, o que possibilita o início do desenvolvimento do diagrama da FTA. Os aspectos do modelo *Viewpoint DSL* a ser criado são mostrados na Figura 3.2. Os aspectos podem ser descritos resumidamente da seguinte forma (LANGLOIS, 2015):

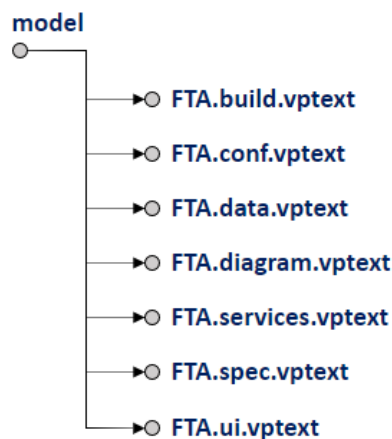


FIGURA 3.2 – Elementos que compõem o modelo.

- ***Spec***

Definição dos aspectos do viewpoint.

- ***Data***

Definição das classes representativas dos elementos do diagrama, dos seus atributos e das associações entre elas.

- ***Diagram***

Cada diagrama possui:

- Contexto (*Context*): no caso do Capella, pode ser uma arquitetura física ou lógica, por exemplo.
- Mapeamento (*Mapping*): Responsável por criar a representação gráfica das metaclasses, atribuindo a elas um rótulo (*label*) e um estilo (*style*).
- Ações (*Actions*): Ações baseadas no mapeamento, como criação e exclusão dos elementos do diagrama.

- ***UI***

Definição da interface gráfica com as propriedades e atributos cujos valores serão inseridos pelos usuários.

- ***Services***

Definição dos serviços (*services*) que orquestrarão a execução do código.

- ***Build***

Especificação do endereço do software Eclipse onde a aplicação será construída em tempo de execução e configuração do endereço do repositório da aplicação.

- ***Configuration***

Estabelece configurações de projeto relacionadas à geração do viewpoint, como por exemplo, o destino da aplicação e as configurações dos dados e dos diagramas.

4 Resultados e Discussão

Para ilustrar a proposta do diagrama da FTA a ser desenvolvido no Kitalpha, foram feitos através da ferramenta Eclipse Obeo Designer os diagramas que esquematizam uma FTA em geral (Figura 4.1) e os seus elementos constituintes: Eventos (Figura 4.2) e Portas Lógicas (Figura 4.3). Cada Diagrama possui um nome associado. Cada Evento possui um símbolo e é composto por uma descrição e pela respectiva probabilidade de ocorrência (probabilidade de falha). Cada Porta Lógica possui um símbolo e uma descrição opcional. Os símbolos dos eventos e portas lógicas foram desenvolvidos de acordo com a norma internacional IEC 61025.

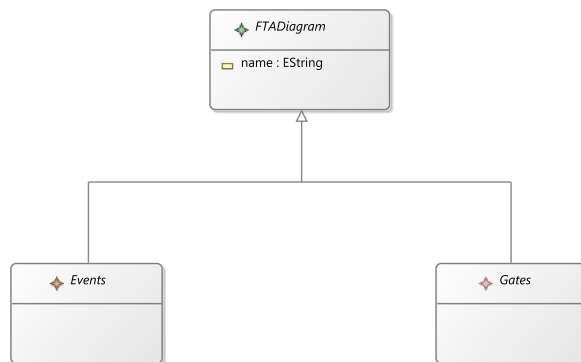


FIGURA 4.1 – Diagrama da FTA.

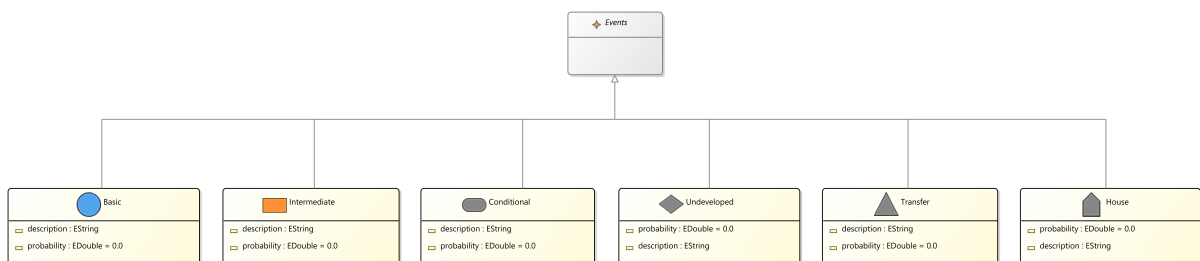


FIGURA 4.2 – Eventos da FTA.

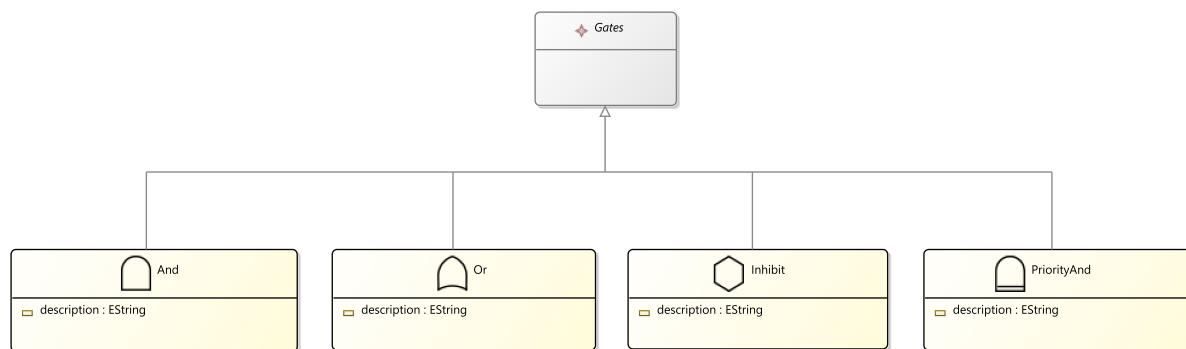
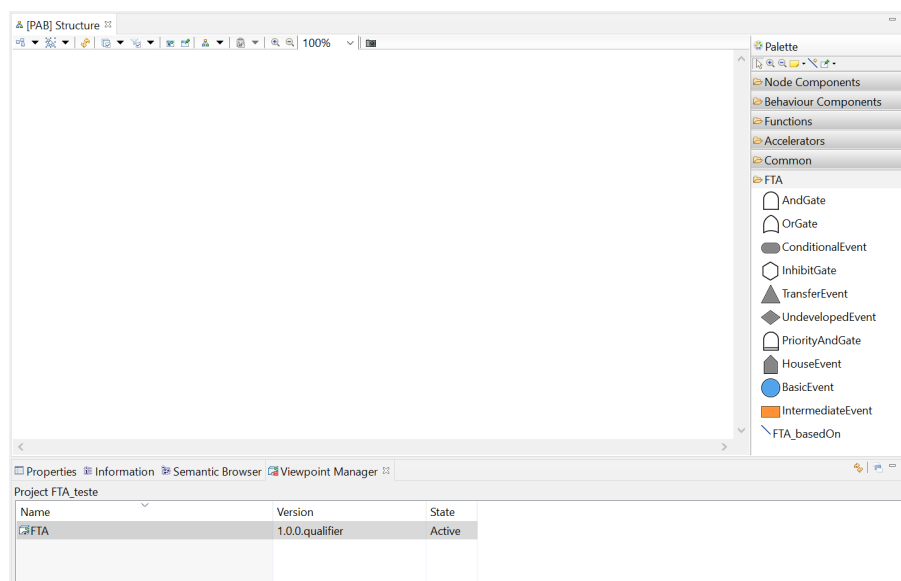


FIGURA 4.3 – Portas Lógicas da FTA.

4.1 Desenvolvimento do Viewpoint e Integração MBSE

O Apêndice A apresenta os códigos desenvolvidos em cada aspecto na criação do *Viewpoint*. Para se criar o viewpoint e gerar o seu pacote a ser utilizado como uma extensão (*Add-on*) do Capella, no ambiente da aba *FTA.spec.vptext*, clica-se com o botão direito do *mouse* e seleciona-se *Generate and package viewpoint*. O pacote gerado é então passado para a pasta *dropins* presente no endereço de projeto do Capella. Para ser utilizado no ambiente do Capella, seleciona-se o caminho *Window* → *Show View* → *Other* → *Kitalpha* → *Viewpoint Manager*. Com a janela do *Viewpoint Manager* aberta, ativa-se o *viewpoint* da FTA ao clicar-se com o botão direito do *mouse* e selecionar-se *Reference* na linha do *viewpoint*. Depois, para se utilizar o diagrama da FTA no contexto de um Componente de Arquitetura Física, basta selecioná-lo ao clicar no botão *Layers* da barra de navegação superior. A Figura 4.4 mostra o *viewpoint* da FTA integrado e disponível em um ambiente de Arquitetura Física do Capella.

FIGURA 4.4 – Integração MBSE do *viewpoint* da FTA.

4.2 Interface Gráfica

As Figuras 4.5 e 4.6 mostram a interface de usuário para os eventos e portas lógicas, respectivamente. Para os eventos, o usuário pode entrar com a descrição e a probabilidade correspondente (opcional). Para as portas lógicas, o usuário pode entrar com alguma descrição. As entradas fornecidas pelo usuário são mostradas na borda do ícone do evento ou porta correspondente.

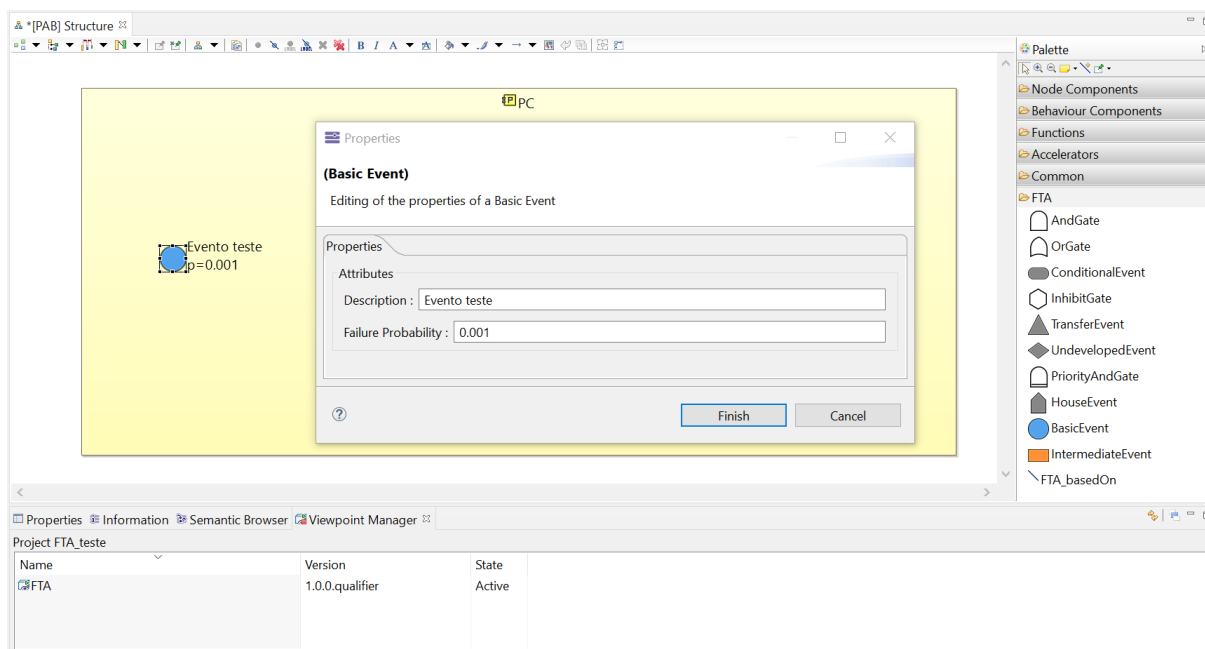


FIGURA 4.5 – Interface para Eventos.

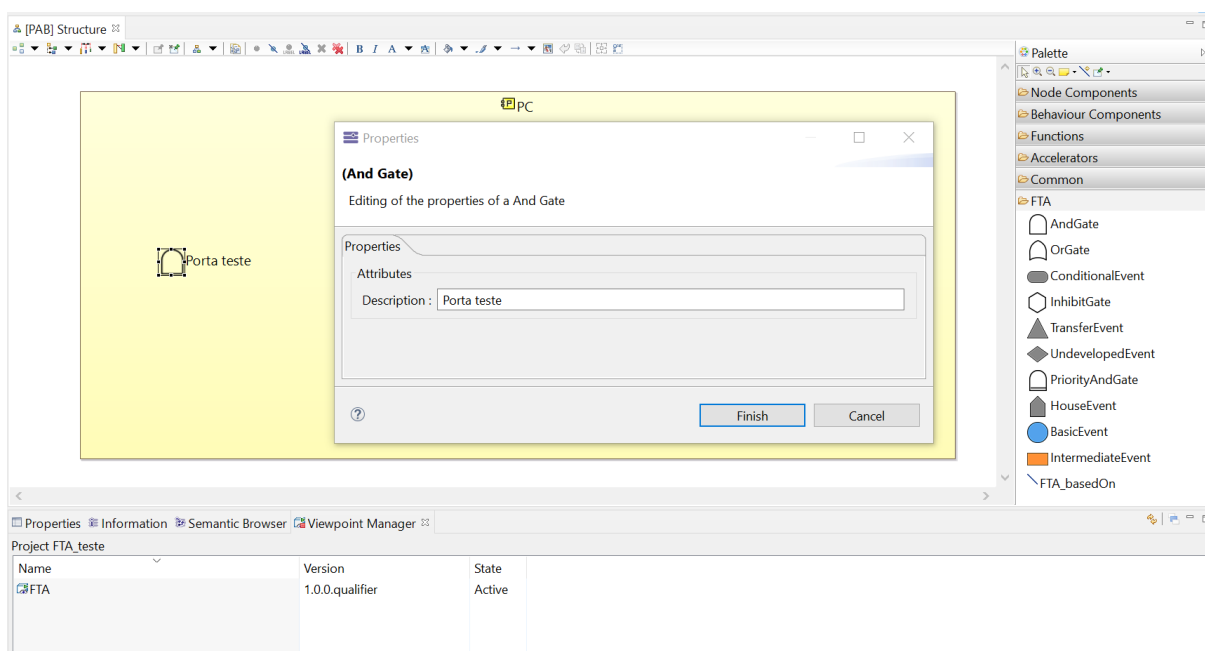


FIGURA 4.6 – Interface para Portas Lógicas.

4.3 Aplicações

Para exemplificar alguns casos possíveis de aplicação do *viewpoint* construído, serão desenvolvidas as Árvores de Falhas para a explosão da câmara de combustão e a perda de navegação de um foguete, bem como um diagrama combinado para a falha no cumprimento da missão espacial (transporte de uma carga paga).

4.3.1 Explosão da Câmara de Combustão

A Figura 4.7 mostra a FTA para a explosão da câmara de combustão do foguete. Nesse caso, como as probabilidades não foram especificadas, a análise do diagrama se restringe a uma análise qualitativa, a qual permite determinar as causas dos eventos indesejados. Pelo diagrama, a explosão ocorre por conta da ocorrência de pelo menos um dos eventos: vazamento de oxigênio, rompimento da câmara de combustão ou derretimento da tubeira.

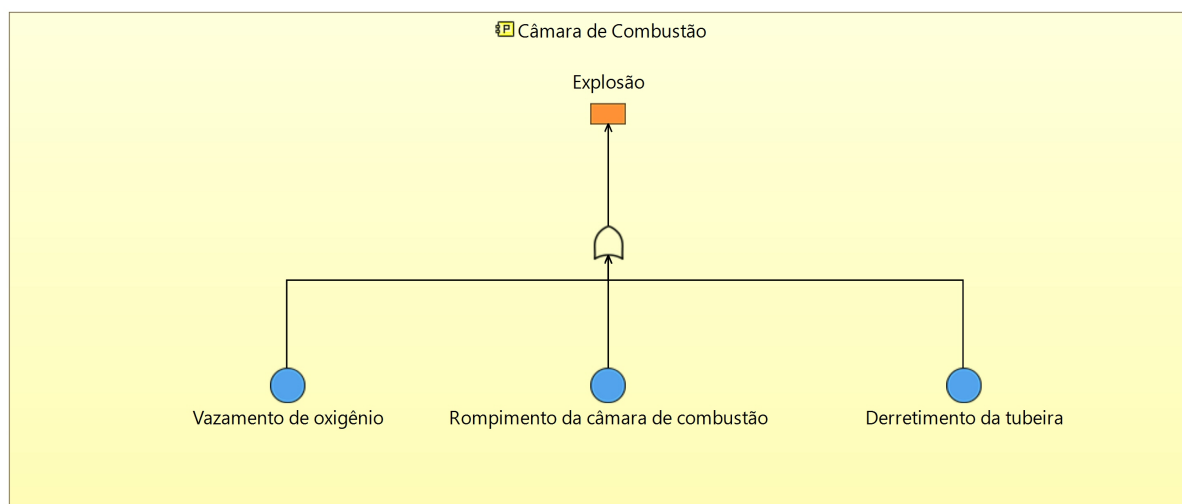


FIGURA 4.7 – FTA para a explosão do foguete.

A especificação das probabilidades através da interface de usuário, Figura 4.8, permite a realização da análise quantitativa, a qual determina a probabilidade de ocorrência dos eventos. Através do diagrama, percebe-se que a probabilidade das causas raízes, quando combinadas por uma porta Or, geram uma probabilidade de falha para o evento de saída de 0.0015.

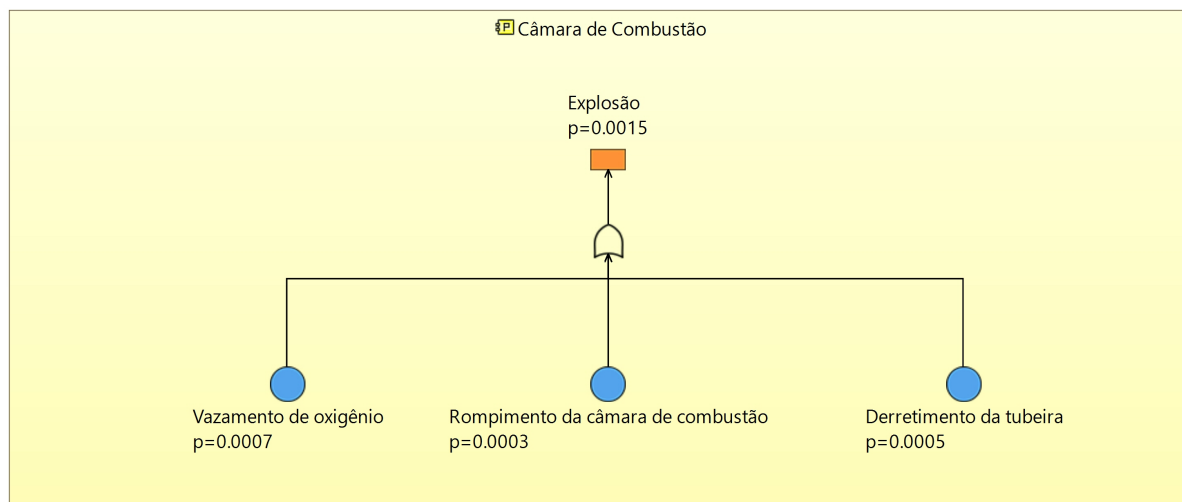


FIGURA 4.8 – FTA da explosão com probabilidades preenchidas.

4.3.2 Perda de Navegação

A Figura 4.9 mostra a FTA construída para o sistema de navegação e controle. Pelo diagrama, percebe-se que o sistema falha (perda de navegação) quando pelo menos um dos seguintes eventos ocorrem: Perda de potência elétrica, Baixa velocidade para o voo controlado, Falha no GPS ou Comandos do software não funcionam. Através do diagrama, percebe-se que a probabilidade de falha para o sistema é de 0.0003.

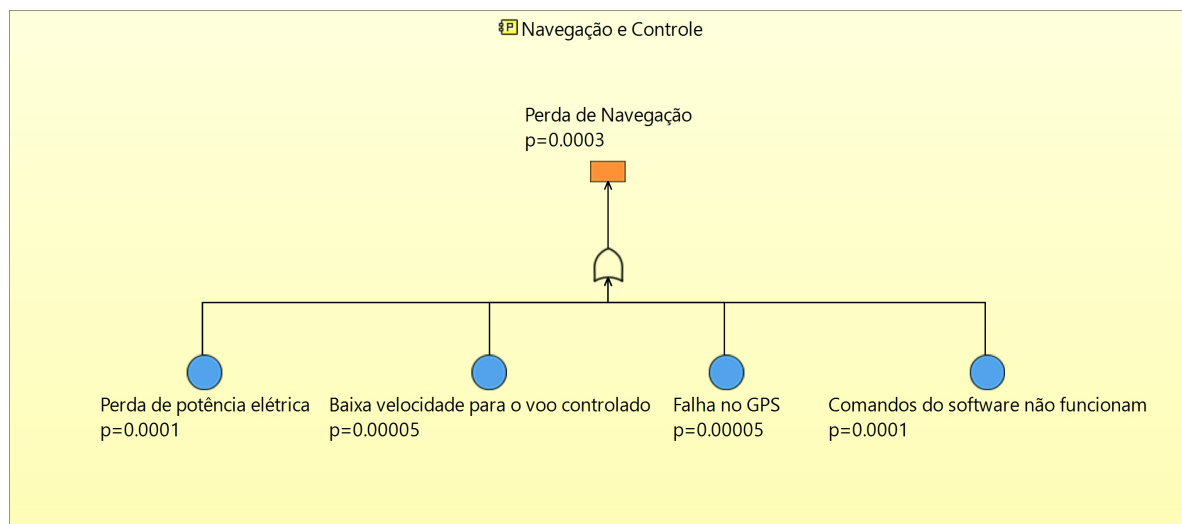


FIGURA 4.9 – FTA para perda de navegação.

4.3.3 Falha da Missão

A Figura 4.10 mostra o diagrama da FTA para a falha da missão espacial do foguete. Percebe-se inicialmente que esse diagrama é composto pelos dois eventos intermediários

desenvolvidos anteriormente: Explosão da Câmara de Combustão e Perda de Navegação. As FTAs para ambos os eventos intermediários também são mostradas no diagrama geral. Além deles, há ainda um evento de transferência para a falha decorrente de um acidente com a carga paga, o que acarretaria também falha da missão. Esse evento de transferência representa um evento que foi desenvolvido em outra parte e que foi apenas referenciado no diagrama geral. Do diagrama, a falha da missão ocorre por conta da ocorrência de pelo menos um dos eventos indesejados: Explosão da Câmara de Combustão, Perda de Navegação ou acidente com a carga paga. Quando combinadas, as probabilidades de falhas dos eventos intermediários e a do evento de transferência resultam em uma probabilidade de 0.0028 para a falha da missão.

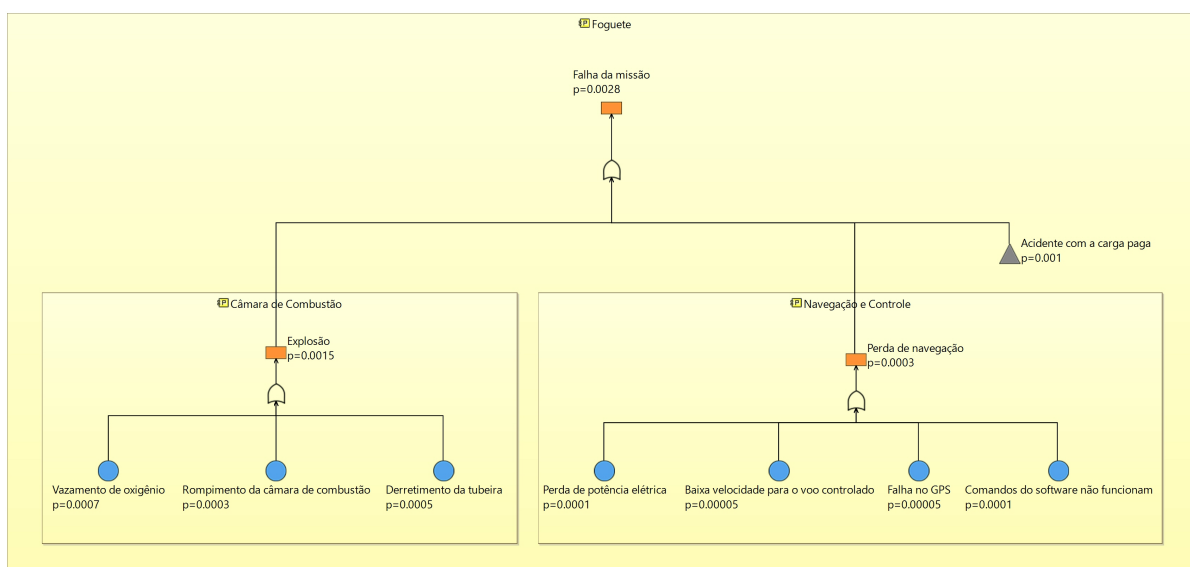


FIGURA 4.10 – FTA para a missão do foguete.

5 Conclusão

O presente trabalho desenvolveu a Análise de Árvore de Falhas (FTA) através de diagramas construídos nos moldes da norma internacional IEC 61025. O desenvolvimento do diagrama da FTA foi feito através de um *viewpoint* no ambiente Kitalpha e o produto obtido foi então integrado à ferramenta MBSE Capella, passando a ser uma nova funcionalidade da ferramenta.

Por ser uma versão simplificada, mas formal, com padronização de uso e de representação gráfica dos eventos, portas lógicas e ramos, bem como de fácil entendimento e utilização pelos usuários, o produto obtido se apresenta como um modelo inicial para a Análise da Árvore de Falhas, o que foi atestado pelas aplicações iniciais desenvolvidas no trabalho.

O principal desafio encontrado na realização deste trabalho foi o de desenvolver a aplicação no Kitalpha com base na pouca experiência pessoal alinhada com poucas referências encontradas na literatura/internet. Muitas vezes, para se implementar uma ideia, foi preciso explorar muito sobre a linguagem DSL da ferramenta, já que não existe uma documentação específica sobre ela, o que normalmente acarretava testar diversas possibilidades para se encontrar uma solução/caminho. Isso possibilitou um grande avanço de conhecimento sobre a ferramenta. Mesmo assim, sem as poucas referências encontradas, este trabalho não seria possível. Além desse desafio, destacam-se também, de forma mais sucinta: entendimento de como um diagrama é estruturado no Kitalpha; entendimento da linguagem DSL utilizada no *viewpoint*.

Por se tratar de uma aplicação bem sucedida e que explorou muito as funcionalidades de uma ferramenta ainda pouco explorada, este trabalho pode ser utilizado como base para muitas outras aplicações que envolvam o desenvolvimento de *viewpoints* no Kitalpha e integração MBSE no Capella de modelos baseados em diagramas para a engenharia de sistemas.

Referências

- BOUDJENNAH, C.; COMBEMALE, B.; EXERTIER, D.; LACRAMPE, S.; PERALDI-FRATI, M.-A. Clarity: Open-sourcing the model-based systems engineering solution capella. 01 2015.
- CALIÒ, E.; GIORGIO, F. D.; PASQUINELLI, M. Deploying model-based systems engineering in thales alenia space italia. *In: CIISE. Proceedings* [...]. [S.l.: s.n.], 2016. p. 112–118.
- COMMITTEE, I. . T. *et al.* Fault tree analysis (fta). **IEC 61025**, IEC, 2006.
- HART, L. E. Introduction to model-based system engineering (mbse) and sysml. *In: RAMBLEWOOD COUNTRY CLUB MOUNT LAUREL, NEW JERSEY. Delaware Valley INCOSE Chapter Meeting. Proceedings* [...]. [S.l.: s.n.], 2015. v. 30.
- INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities. [S.l.]: Wiley, 2015. ISBN 9781119015123.
- KITALPHA. 2021. Disponível em: <https://www.polarsys.org/projects/polarsys.kitalpha>. Acessado em: 20 jun. 2021.
- LANGLOIS, B. **TEXTUAL DSL FOR ARCHITECTURE DESCRIPTION**. 2015. Disponível em: https://wiki.eclipse.org/images/2/26/Kitalpha-B02-AF_and_Viewpoint_DSLs.pdf. Acessado em: 08 nov. 2021.
- LIBRARY, S. S.; ADMINISTRATION, N. **NASA Systems Engineering Handbook - NASA SP-2016-6105 Rev2: Design Test Integrate Fly**. [S.l.]: CreateSpace Independent Publishing Platform, 2017. ISBN 9781977821966.
- MCDERMOTT, T.; HUTCHISON, N.; CLIFFORD, M.; AKEN, E. V.; SALADO, A.; HENDERSON, K. Benchmarking the benefits and current maturity of model-based systems engineering across the enterprise: Results of the mbse maturity survey. 03 2020.
- ROQUES, P. Reminders for the arcadia method. *In: . Proceedings* [...]. [S.l.: s.n.], 2018.
- Wikipedia contributors. **Fault tree analysis — Wikipedia, The Free Encyclopedia**. 2020. Disponível em: https://en.wikipedia.org/wiki/Fault_tree_analysis. Acessado em: 23 jun. 2021.

Wikipedia contributors. **Capella (engineering)** — **Wikipedia, The Free Encyclopedia**. 2021. Disponível em: [https://en.wikipedia.org/wiki/Capella_\(engineering\)](https://en.wikipedia.org/wiki/Capella_(engineering)). Acessado em: 20 jun. 2021.

Apêndice A - Códigos do Viewpoint

A.1 Aspects

```
1  /**
2  * Copyright (c) PolarSys, 2021. All rights reserved.
3  *
4  * Viewpoint FTA
5  * @author: Geune Vieira Quintino
6  * @date: 19/10/2021
7  *
8  */
9
10 Viewpoint FTA {
11     name: "FTA"
12     Data FTA.data
13     UI FTA.ui
14     Diagrams FTA.diagram
15     Services FTA.services
16     Build FTA.build
17     Configuration FTA.conf
18 }
```

A.2 Data

```
1  /**
2  * Copyright (c) PolarSys, 2021. All rights reserved.
3  *
4  * Viewpoint FTA
5  * @author: Geune Vieira Quintino
6  * @date: 19/10/2021
7  *
8  */
9
10 Data FTA.data {
11   Class AbstractComponent {
12     superClass external emde.ExtensibleElement
13     abstract: true
14     Associations:
15       basedOn refers [0,*] AbstractComponent
16   }
17
18   Class AndGate {
19     icon: "and.png"
20     extends la.LogicalComponent, pa.PhysicalComponent
21     superClass AbstractComponent
22     Attributes:
23       ^description type ecore.EString
24   }
25
26   Class OrGate {
27     icon: "or.png"
28     extends la.LogicalComponent, pa.PhysicalComponent
29     superClass AbstractComponent
30     Attributes:
31       ^description type ecore.EString
32   }
33
34   Class ConditionalEvent {
35     icon: "conditional.png"
36     extends la.LogicalComponent, pa.PhysicalComponent
37     superClass AbstractComponent
38     Attributes:
39       ^description type ecore.EString
40       probability type ecore.EString
41   }
42
43   Class InhibitGate {
44     icon: "inhibit.png"
45     extends la.LogicalComponent, pa.PhysicalComponent
46     superClass AbstractComponent
47     Attributes:
48       ^description type ecore.EString
49   }
50
51   Class PriorityAndGate {
52     icon: "priorand.png"
53     extends la.LogicalComponent, pa.PhysicalComponent
54     superClass AbstractComponent
55     Attributes:
56       ^description type ecore.EString
```

```
57     }
58
59     Class TransferEvent {
60         icon: "transfer.png"
61         extends la.LogicalComponent, pa.PhysicalComponent
62         superClass AbstractComponent
63         Attributes:
64             ^description type ecore.EString
65             probability type ecore.EString
66     }
67
68     Class UndevelopedEvent {
69         icon: "undeveloped.png"
70         extends la.LogicalComponent, pa.PhysicalComponent
71         superClass AbstractComponent
72         Attributes:
73             ^description type ecore.EString
74             probability type ecore.EString
75     }
76
77     Class HouseEvent {
78         icon: "house.png"
79         extends la.LogicalComponent, pa.PhysicalComponent
80         superClass AbstractComponent
81         Attributes:
82             ^description type ecore.EString
83             probability type ecore.EString
84     }
85
86     Class BasicEvent {
87         icon: "basic.png"
88         extends la.LogicalComponent, pa.PhysicalComponent
89         superClass AbstractComponent
90         Attributes:
91             ^description type ecore.EString
92             probability type ecore.EString
93     }
94
95     Class IntermediateEvent {
96         icon: "intermediate.png"
97         extends la.LogicalComponent, pa.PhysicalComponent
98         superClass AbstractComponent
99         Attributes:
100             ^description type ecore.EString
101             probability type ecore.EString
102     }
103 }
```

A.3 Diagram

```
1  /**
2  * Copyright (c) PolarSys, 2021. All rights reserved.
3  *
4  * Viewpoint FTA
5  * @author: Geune Vieira Quintino
6  * @date: 19/10/2021
7  *
8  */
9
10 Diagrams FTA.diagram {
11   DiagramExtension "diagramExtension" {
12     extended-diagram: PhysicalArchitectureBlank //extended diagram
13     Mapping {
14       Container FTAContainer {
15         import: PAB_PC //import a container
16         Contains {
17           Node AndGate {
18             domain-context: FTA.data.AndGate
19             provided-by association external emde.ExtensibleElement.ownedExtensions
20             Representation {
21               Label {
22                 content: FTA.data.AndGate.^description police: black position:
23                   border alignment: left
24               }
25               Style {
26                 Image {path:"and.png"}
27               }
28             }
29           }
30           Node OrGate {
31             domain-context: FTA.data.OrGate
32             provided-by association external emde.ExtensibleElement.ownedExtensions
33             Representation {
34               Label {
35                 content: FTA.data.OrGate.^description police: black position:
36                   border alignment: left
37               }
38               Style {
39                 Image {path:"or.png"}
40               }
41             }
42           }
43           Node ConditionalEvent {
44             domain-context: FTA.data.ConditionalEvent
45             provided-by association external emde.ExtensibleElement.ownedExtensions
46             Representation {
47               Label {
48                 content: FTA.data.ConditionalEvent.^description police: black
49                   position: border alignment: left
50               }
51               Style {
52                 Image {path:"conditional.png"}
53               }
54             }
55           }
56         }
57       }
58     }
59   }
60 }
```

```
54         Representation {
55             condition: Java whenProbNotEmptyConditionalEvent
56             Label {
57                 content: FTA.data.ConditionalEvent.^description + "\r\np=" +
                    FTA.data.ConditionalEvent.probability police: black
                    position: border alignment: left
58             }
59             Style {
60                 Image {path:"conditional.png"}
61             }
62         }
63     }
64
65     Node InhibitGate {
66         domain-context: FTA.data.InhibitGate
67         provided-by association external emde.ExtensibleElement.ownedExtensions
68         Representation {
69             Label {
70                 content: FTA.data.InhibitGate.^description police: black
                    position: border alignment: left
71             }
72             Style {
73                 Image {path:"inhibit.png"}
74             }
75         }
76     }
77
78     Node PriorityAndGate {
79         domain-context: FTA.data.PriorityAndGate
80         provided-by association external emde.ExtensibleElement.ownedExtensions
81         Representation {
82             Label {
83                 content: FTA.data.PriorityAndGate.^description police: black
                    position: border alignment: left
84             }
85             Style {
86                 Image {path:"priorand.png"}
87             }
88         }
89     }
90
91     Node TransferEvent {
92         domain-context: FTA.data.TransferEvent
93         provided-by association external emde.ExtensibleElement.ownedExtensions
94         Representation {
95             Label {
96                 content: FTA.data.TransferEvent.^description police: black
                    position: border alignment: left
97             }
98             Style {
99                 Image {path:"transfer.png"}
100             }
101         }
102         Representation {
103             condition: Java whenProbNotEmptyTransferEvent
104             Label {
105                 content: FTA.data.TransferEvent.^description + "\r\np=" +
                    FTA.data.TransferEvent.probability police: black position:
                    border alignment: left
```

```
106     }
107     Style {
108         Image {path:"transfer.png"}
109     }
110 }
111 }
112
113 Node UndevelopedEvent {
114     domain-context: FTA.data.UndevelopedEvent
115     provided-by association external emde.ExtensibleElement.ownedExtensions
116     Representation {
117         Label {
118             content: FTA.data.UndevelopedEvent.^description police: black
119             position: border alignment: left
120         }
121         Style {
122             Image {path:"undeveloped.png"}
123         }
124     }
125     Representation {
126         condition: Java whenProbNotEmptyUndevelopedEvent
127         Label {
128             content: FTA.data.UndevelopedEvent.^description + "\r\np=" +
129                 FTA.data.UndevelopedEvent.probability police: black
130             position: border alignment: left
131         }
132         Style {
133             Image {path:"undeveloped.png"}
134         }
135     }
136 }
137
138 Node HouseEvent {
139     domain-context: FTA.data.HouseEvent
140     provided-by association external emde.ExtensibleElement.ownedExtensions
141     Representation {
142         Label {
143             content: FTA.data.HouseEvent.^description police: black
144             position: border alignment: left
145         }
146         Style {
147             Image {path:"house.png"}
148         }
149     }
150     Representation {
151         condition: Java whenProbNotEmptyHouseEvent
152         Label {
153             content: FTA.data.HouseEvent.^description + "\r\np=" +
154                 FTA.data.HouseEvent.probability police: black position:
155                 border alignment: left
156         }
157         Style {
158             Image {path:"house.png"}
159         }
160     }
161 }
162
163 Node BasicEvent {
164     domain-context: FTA.data.BasicEvent
```

```

159         provided-by association external emde.ExtensibleElement.ownedExtensions
160 Representation {
161     Label {
162         content: FTA.data.BasicEvent.^description police: black
163         position: border alignment: left
164     }
165     Style {
166         Image {path:"basic.png"}
167     }
168 Representation {
169     condition: Java whenProbNotEmptyBasicEvent
170     Label {
171         content: FTA.data.BasicEvent.^description + "\r\np=" +
172             FTA.data.BasicEvent.probability police: black position:
173             border alignment: left
174     }
175     Style {
176         Image {path:"basic.png"}
177     }
178 }
179 Node IntermediateEvent {
180     domain-context: FTA.data.IntermediateEvent
181     provided-by association external emde.ExtensibleElement.ownedExtensions
182 Representation {
183     Label {
184         content: FTA.data.IntermediateEvent.^description police: black
185         position: border alignment: left
186     }
187     Style {
188         Image {path:"intermediate.png"}
189     }
190 Representation {
191     condition: Java whenProbNotEmptyIntermediateEvent
192     Label {
193         content: FTA.data.IntermediateEvent.^description + "\r\np=" +
194             FTA.data.IntermediateEvent.probability police: black
195         position: border alignment: left
196     }
197     Style {
198         Image {path:"intermediate.png"}
199     }
200 }
201 }
202
203 Edge FTA_basedOn {
204     association-context: FTA.data.AbstractComponent.basedOn
205     source: FTAContainer.AndGate, FTAContainer.OrGate,
206         FTAContainer.ConditionalEvent,
207         FTAContainer.InhibitGate, FTAContainer.PriorityAndGate,
208         FTAContainer.TransferEvent,
209         FTAContainer.UndevelopedEvent, FTAContainer.HouseEvent,
210         FTAContainer.BasicEvent,
211         FTAContainer.IntermediateEvent

```

```

209         target: FTAContainer.AndGate, FTAContainer.OrGate,
210             FTAContainer.ConditionalEvent,
211             FTAContainer.InhibitGate, FTAContainer.PriorityAndGate,
212             FTAContainer.TransferEvent,
213             FTAContainer.UndevelopedEvent, FTAContainer.HouseEvent,
214             FTAContainer.BasicEvent,
215             FTAContainer.IntermediateEvent
216     Representation {
217         Style {
218             end-decorator: InputArrow
219             color: black
220         }
221     }
222 }
223
224 Actions {
225     /* AndGate Actions */
226     Create AndGate_CT { label: "AndGate" action-for: FTAContainer.AndGate }
227     Drop AndGate_DR { action-for: FTAContainer.AndGate }
228     Delete AndGate_DL { action-for: FTAContainer.AndGate }
229
230     /* OrGate Actions */
231     Create OrGate_CT { label: "OrGate" action-for: FTAContainer.OrGate }
232     Drop OrGate_DR { action-for: FTAContainer.OrGate }
233     Delete OrGate_DL { action-for: FTAContainer.OrGate }
234
235     /* ConditionalEvent Actions */
236     Create ConditionalEvent_CT { label: "ConditionalEvent" action-for:
237         FTAContainer.ConditionalEvent }
238     Drop ConditionalEvent_DR { action-for: FTAContainer.ConditionalEvent }
239     Delete ConditionalEvent_DL { action-for: FTAContainer.ConditionalEvent }
240
241     /* InhibitGate Actions */
242     Create InhibitGate_CT { label: "InhibitGate" action-for:
243         FTAContainer.InhibitGate }
244     Drop InhibitGate_DR { action-for: FTAContainer.InhibitGate }
245     Delete InhibitGate_DL { action-for: FTAContainer.InhibitGate }
246
247     /* TransferEvent Actions */
248     Create TransferEvent_CT { label: "TransferEvent" action-for:
249         FTAContainer.TransferEvent }
250     Drop TransferEvent_DR { action-for: FTAContainer.TransferEvent }
251     Delete TransferEvent_DL { action-for: FTAContainer.TransferEvent }
252
253     /* UndevelopedEvent Actions */
254     Create UndevelopedEvent_CT { label: "UndevelopedEvent" action-for:
255         FTAContainer.UndevelopedEvent }
256     Drop UndevelopedEvent_DR { action-for: FTAContainer.UndevelopedEvent }
257     Delete UndevelopedEvent_DL { action-for: FTAContainer.UndevelopedEvent }
258
259     /* PriorityAndGate Actions */
260     Create PriorityAndGate_CT { label: "PriorityAndGate" action-for:
261         FTAContainer.PriorityAndGate }
262     Drop PriorityAndGate_DR { action-for: FTAContainer.PriorityAndGate }
263     Delete PriorityAndGate_DL { action-for: FTAContainer.PriorityAndGate }
264
265     /* HouseEvent Actions */
266     Create HouseEvent_CT { label: "HouseEvent" action-for: FTAContainer.HouseEvent }

```



```
260     Drop HouseEvent_DR { action-for: FTAContainer.HouseEvent }
261     Delete HouseEvent_DL { action-for: FTAContainer.HouseEvent }
262
263     /* BasicEvent Actions */
264     Create BasicEvent_CT { label: "BasicEvent" action-for: FTAContainer.BasicEvent }
265     Drop BasicEvent_DT { action-for: FTAContainer.BasicEvent }
266     Delete BasicEvent_DL { action-for: FTAContainer.BasicEvent }
267
268     /* IntermediateEvent Actions */
269     Create IntermediateEvent_CT { label: "IntermediateEvent" action-for:
270         FTAContainer.IntermediateEvent }
271     Drop IntermediateEvent_DT { action-for: FTAContainer.IntermediateEvent }
272     Delete IntermediateEvent_DL { action-for: FTAContainer.IntermediateEvent }
273
274     /*FTA_basedOn Actions */
275     Create FTA_basedOn_CT { label: "FTA_basedOn" action-for: FTA_basedOn }
276     Delete FTA_basedOn_DT { action-for: FTA_basedOn }
277     ReconnectEdge FTA_basedOn_RET { action-for: FTA_basedOn }
278 }
279 }
```

A.4 UI

```
1  /**
2  * Copyright (c) PolarSys, 2021. All rights reserved.
3  *
4  * Viewpoint FTA.ui
5  * @author: Geune Vieira Quintino
6  * @date: 19/10/2021
7  *
8  */
9
10 UIDescription FTA.ui {
11     UI FTA_AndGate {
12         label: "Properties"
13         Container FTA_AndGate_Section {
14             Container FTA_AndGate_AttributeGroup {
15                 label: "Attributes"
16                 Field descriptionFieldAndGate label: "Description" type text, mapped-to
17                     FTA.data.AndGate.^description
18             }
19         }
20     }
21     UI FTA_OrGate {
22         label: "Properties"
23         Container FTA_OrGate_Section {
24             Container FTA_OrGate_AttributeGroup {
25                 label: "Attributes"
26                 Field descriptionFieldOrGate label: "Description" type text, mapped-to
27                     FTA.data.OrGate.^description
28             }
29         }
30     }
31     UI FTA_ConditionalEvent {
32         label: "Properties"
33         Container FTA_ConditionalEvent_Section {
34             Container FTA_ConditionalEvent_AttributeGroup {
35                 label: "Attributes"
36                 Field descriptionFieldConditionalEvent label: "Description" type text,
37                     mapped-to FTA.data.ConditionalEvent.^description
38                 Field probabilityConditionalEvent label: "Failure Probability" type text,
39                     mapped-to FTA.data.ConditionalEvent.probability
40             }
41         }
42     }
43     UI FTA_InhibitGate {
44         label: "Properties"
45         Container FTA_InhibitGate_Section {
46             Container FTA_InhibitGate_AttributeGroup {
47                 label: "Attributes"
48                 Field descriptionFieldInhibitGate label: "Description" type text, mapped-to
49                     FTA.data.InhibitGate.^description
50             }
51         }
52     }
53 }
```

```
52     UI FTA_PriorityAndGate {
53         label: "Properties"
54         Container FTA_PriorityAndGate_Section {
55             Container FTA_PriorityAndGate_AttributeGroup {
56                 label: "Attributes"
57                 Field descriptionFieldPriorityAndGate label: "Description" type text,
                    mapped-to FTA.data.PriorityAndGate.^description
58             }
59         }
60     }
61
62     UI FTA_TransferEvent {
63         label: "Properties"
64         Container FTA_TransferEvent_Section {
65             Container FTA_TransferEvent_AttributeGroup {
66                 label: "Attributes"
67                 Field descriptionFieldTransferEvent label: "Description" type text,
                    mapped-to FTA.data.TransferEvent.^description
68                 Field probabilityTransferEvent label: "Failure Probability" type text,
                    mapped-to FTA.data.TransferEvent.probability
69             }
70         }
71     }
72
73     UI FTA_UndevelopedEvent {
74         label: "Properties"
75         Container FTA_UndevelopedEvent_Section {
76             Container FTA_UndevelopedEvent_AttributeGroup {
77                 label: "Attributes"
78                 Field descriptionFieldUndevelopedEvent label: "Description" type text,
                    mapped-to FTA.data.UndevelopedEvent.^description
79                 Field probabilityUndevelopedEvent label: "Failure Probability" type text,
                    mapped-to FTA.data.UndevelopedEvent.probability
80             }
81         }
82     }
83
84     UI FTA_HouseEvent {
85         label: "Properties"
86         Container FTA_HouseEvent_Section {
87             Container FTA_UndevelopedEvent_AttributeGroup {
88                 label: "Attributes"
89                 Field descriptionFieldHouseEvent label: "Description" type text, mapped-to
                    FTA.data.HouseEvent.^description
90                 Field probabilityHouseEvent label: "Failure Probability" type text,
                    mapped-to FTA.data.HouseEvent.probability
91             }
92         }
93     }
94
95     UI FTA_BasicEvent {
96         label: "Properties"
97         Container FTA_BasicEvent_Section {
98             Container FTA_BasicEvent_AttributeGroup {
99                 label: "Attributes"
100                Field descriptionFieldBasicEvent label: "Description" type text, mapped-to
                    FTA.data.BasicEvent.^description
101                Field probabilityBasicEvent label: "Failure Probability" type text,
                    mapped-to FTA.data.BasicEvent.probability
```

```
102     }
103   }
104 }
105
106 UI FTA_IntermediateEvent {
107   label: "Properties"
108   Container FTA_IntermediateEvent_Section {
109     Container FTA_IntermediateEvent_AttributeGroup {
110       label: "Attributes"
111       Field descriptionFieldOutputEvent label: "Description" type text, mapped-to
112         FTA.data.IntermediateEvent.^description
113       Field probabilityOutputEvent label: "Failure Probability" type text,
114         mapped-to FTA.data.IntermediateEvent.probability
115     }
116   }
117 }
118 }
```

A.5 Services

```
1  /**
2   * Copyright (c) PolarSys, 2021. All rights reserved.
3   *
4   * Viewpoint FTA
5   * @author: Geune Vieira Quintino
6   * @date: 19/10/2021
7   *
8   */
9
10 Rules FTA.rules {
11     Rule RuleOne type Java
12     Rule RuleTwo type Java
13
14 }
15
16 Services FTA.services {
17     Service MyService orchestrates RuleOne, RuleTwo
18 }
```

A.6 Configuration

```
1  /**
2  * Copyright (c) PolarSys, 2021. All rights reserved.
3  *
4  * Viewpoint FTA
5  * @author: Geune Vieira Quintino
6  * @date: 19/10/2021
7  *
8  */
9
10 Configuration FTA.conf {
11     target Capella
12     project org.polarsys.capella.vp.fta
13     nsuri "http://www.polarsys.org/capella/FTA"
14     release {
15         version: 1.0.0.qualifier
16         description: "Viewpoint Description"
17         execution environments: "JavaSE-1.6"
18     }
19     generation {
20         data (
21             Model: true
22             Edit: true
23             Editor: false
24             Test: false
25             Javadoc: false
26             OverwriteEcore: true
27         )
28         diagram (
29             OverwriteOdesign: true
30         )
31         documentation (
32             EcoreToHtml: false
33         )
34     }
35 }
```

A.7 Build

```
1  /**
2   * Copyright (c) PolarSys, 2021. All rights reserved.
3   *
4   * Viewpoint FTA
5   * @author: Geune Viera Quintino
6   * @date: 19/10/2021
7   *
8   */
9
10 Build FTA.build {
11     target-platform: "C://eclipse.exe"
12     repository: git "http://shortName/FTA.git"
13     features: org.polarsys.kitalpha.vp.FTA.feature
14 }
```

FOLHA DE REGISTRO DO DOCUMENTO

^{1.} CLASSIFICAÇÃO/TIPO <p style="text-align: center;">TC</p>	^{2.} DATA <p style="text-align: center;">02 de dezembro de 2021</p>	^{3.} REGISTRO N° <p style="text-align: center;">DCTA/ITA/TC-149/2021</p>	^{4.} N° DE PÁGINAS <p style="text-align: center;">46</p>
^{5.} TÍTULO E SUBTÍTULO: <p>Acoplamento da Análise de Árvore de Falhas em uma Arquitetura Sistêmica</p>			
^{6.} AUTOR(ES): <p>Geune Vieira Quintino</p>			
^{7.} INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES): <p>Instituto Tecnológico de Aeronáutica - ITA</p>			
^{8.} PALAVRAS-CHAVE SUGERIDAS PELO AUTOR: <p>MBSE; Viewpoint DSL; Kitalpha; Capella; FTA</p>			
^{9.} PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO: <p>Análise de falhas; Árvore de falhas; Engenharia de sistemas; Engenharia de software; Computação; Engenharia aeroespacial</p>			
^{10.} APRESENTAÇÃO: (X) Nacional () Internacional <p>ITA, São José dos Campos. Curso de Graduação em Engenharia Aeroespacial. Orientador: Christopher Shneider Cerqueira. Publicado em 2021.</p>			
^{11.} RESUMO: <p>Este trabalho irá explorar a utilização da Análise de Árvore de Falhas para avaliação de segurança e confiabilidade de arquiteturas construídas em softwares de Engenharia de Sistemas, de forma a explicitar a robustez de uma arquitetura de um sistema complexo. Os sistemas complexos requerem ferramentas de exploração e gerenciamento de sua complexidade para possibilitar aos engenheiros: a visão do domínio do problema identificando as necessidades dos <i>stakeholders</i> e seus requisitos; a visão do domínio da solução objetivando identificar alternativas de arquiteturas que possam vir a se tornar soluções candidatas à concretização do projeto. Com o auxílio dessa análise integrada, será possível construir a arquitetura em uma ferramenta e explorar a análise de risco no modelo e suas implicações.</p>			
^{12.} GRAU DE SIGILO: <p style="text-align: center;">(X) OSTENSIVO () RESERVADO () SECRETO</p>			